Examen 2. Preguntas y respuestas comentadas

1. Para representar el estado de las camas usamos un tipo abstracto de dato (TAD) con dos subclases (disponible y ocupado).

e) Completamente en desacuerdo.

Un Tipo Abstracto de Dato es un mecanismo software para manejar datos (listas, árboles, fechas, …) sin tener contacto con ellos. El concepto de Tipo Abstracto de Dato carece de sentido en este contexto. Para operar con el estado de un objeto sin preguntar por su valor se utiliza una estructura de clases regida por una clase abstracta que sirve de interfaz al conjunto de clases concretas que detentan el comportamiento de cada estado del objeto en cuestión. Ver patrón Estado.

2. El ratón mueve las figuras que están en la ventana. Por esta causa cada figura debe conocer la posición del ratón, cuando se pulsa, para saber si puede actuar sobre ella.

a) Completamente de acuerdo.

De esta forma el ratón se desentiende de las figuras; solo mantiene una relación indiferente (ambigua) con ellas comunicándoles su posición.

4. El patrón cadena de responsabilidades sirve para:

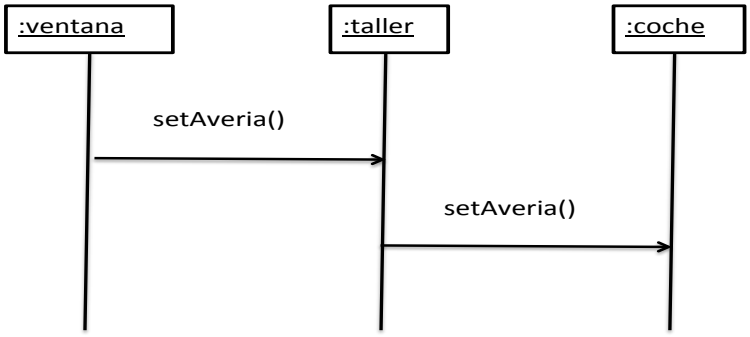
a) Definir la responsabilidad de cada objeto de una lista.
b) Facilitar la modificación de cada objeto de una lista según su responsabilidad.
c) Hacer más ambigua una lista.
d) Conocer con más detalle el manejo de una lista.

a) Falso. La definición de la responsabilidad de cada objeto se hace cuando se definen sus clases respectivas.

b) Falso. La facilidad de modificación de los objetos de la lista está dado por el diseño de sus clases respectivas. El patrón reduce el grado de sorpresa del elemento que utiliza la lista escondiendo la lista a sus ojos.

c) Verdadero. El patrón sirve para reducir el grado de sorpresa del elemento que utiliza la lista; establece una relación ambigua con la lista porque le esconde la lista. El elemento que utiliza la lista ignora que hay una lista, solo conoce al primer miembro de la lista; el resto de la lista queda oculto detrás.

d) *Falso. El propósito del patrón es ocultar la lista por tanto, oculta sus detalles.*



3. El diseño de la figura anterior:

a) No cumple el Principio de Ocultación de Información porque Taller pasa el mismo dato Avería que recibe.

b) No cumple el Principio de Ocultación de Información porque Coche recibe el dato a través de Taller.

c) No cumple el Principio de Ocultación de Información porque Coche es un sumidero donde nada sale

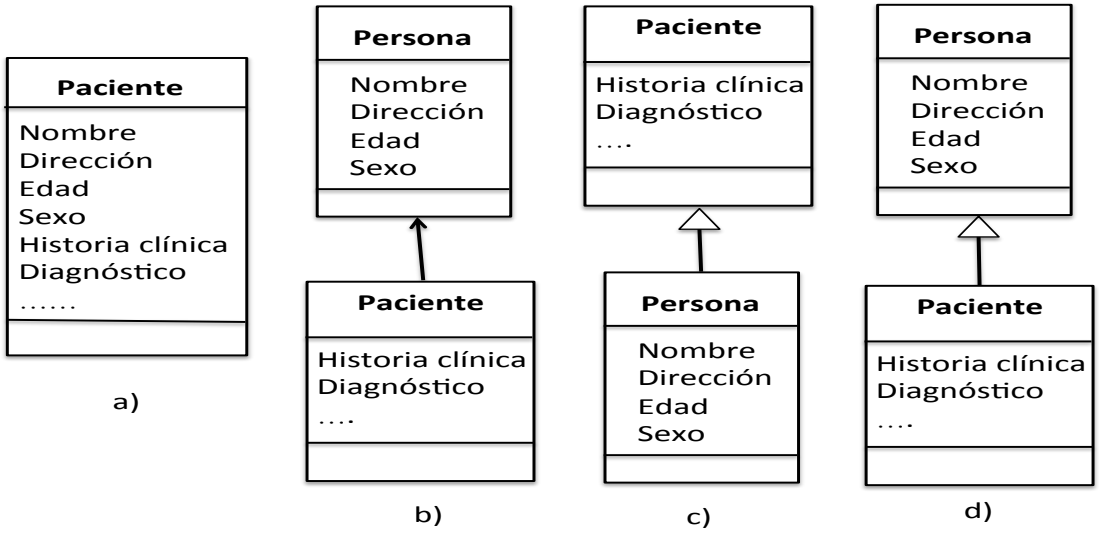
d) Cumple el Principio de Ocultación de Información porque el taller es el encargado de emitir la avería.

a) Falso. Si Taller pasa el mismo dato que recibe solo dice que Taller no cumple la condición de función en el sentido estructurado, pero nada dice respecto al Principio de Ocultación.

b) Verdadero. Se incumple el Principio de Ocultación porque Taller conoce el interior de Coche. La información de la avería debe llegar a Coche desde el exterior mediante una ventana o formulario asociado a Coche.

c) Falso. La condición de sumidero afecta a una función en el sentido estructurado, pero no al Principio de Ocultación.

d) Falso. Incumple el Principio de Ocultación precisamente porque el objeto :taller emite la avería. La idea de representar la realidad en el universo de los objetos software acarrea dificultades. En este caso porque se incumple el Principio de Ocultación y se producen consecuencias desfavorables.



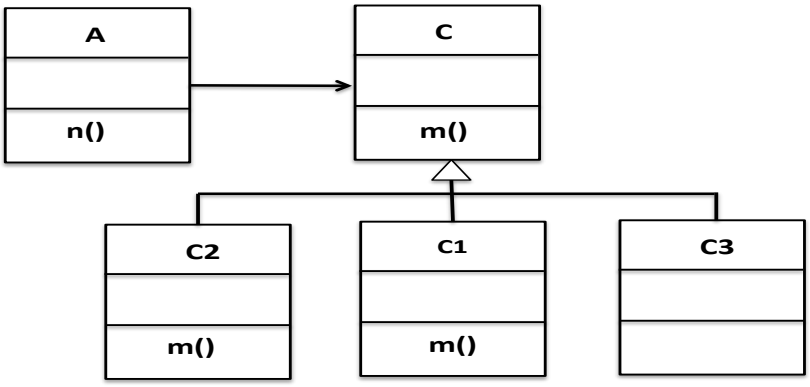
5. ¿Qué diseño de la figura anterior considera mejor?

a) *Este diseño parece el más simple porque tiene menos clases, pero es una falsa imagen. La complejidad de persona y paciente está presente, solo que apelotonado en una clase. Esto impide distribuir trabajo (alguien que se ocupe de persona y alguien que se ocupe de paciente) y además dificulta la modificación y evolución del sistema. Los conceptos de persona y paciente son diferentes, por tanto deben estar separados.*

b) *La solución software adecuada es asociar Paciente con Persona porque expresa mejor la diferencia entre estos conceptos y porque es una relación más débil, más separable que la relación de herencia.*

c) *La relación Persona es un Paciente expresa que toda persona es un paciente, por tanto está equivocado.*

d) *La relación de Paciente "hereda de" Persona parece adecuada en virtud de la interpretación de la herencia software como una relación "es un". Sin embargo, en el presente caso sólo es una relación circunstancial: el paciente es una persona porque se trata de un hospital, pero el concepto de paciente no implica ser persona. Por tanto, es poco conveniente ligarlos tan fuertemente en el software. Resulta mejor una relación se asociación mucho más débil que la herencia. Respuesta b).*



6. El diseño de la figura anterior es:

a) Adecuado para reutilizar m().
b) Inadecuado para extender las alternativas de m().
c) Adecuado para perturbar poco a n().
d) Inadecuado para el polimorfismo de n() de A.

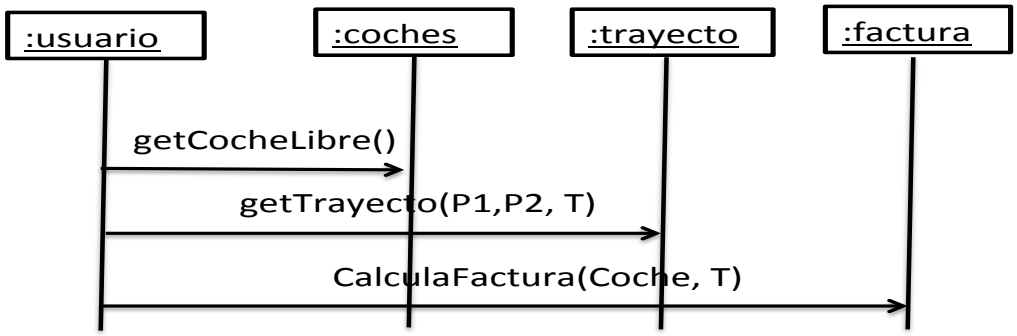
a) Falso. Reutilizar significa usar lo mismo en condiciones diferentes. Los tres métodos m() que aparecen en el diseño son diferentes. Por tanto, no se trata de reutilizar m().

b) Falso. El diseño facilita extender las diversas alternativas de los métodos m() porque el método m() de la superclase introduce la ambigüedad necesaria para ocultar la diversidad de los métodos m() en las subclases.

c) Cierto, porque n() tiene una relación de dependencia indiferente (ambigua) con los métodos m() gracias a la ambigüedad introducida por el método m() de la superclase. Los cambios en los métodos m() no le importan a n().

d) *Falso. El polimorfismo se produce cuando algoritmos diferentes se invocan del mismo modo. Es el caso de los métodos m() en las subclases: sus códigos son diferentes pero se llaman de la misma manera. En el diseño de la figura no tiene sentido el término polimorfismo aplicado a n().*

Un apunte adicional, la clase C3 sobre porque es idéntica a la clase C.



7. El diseño de la figura anterior es:

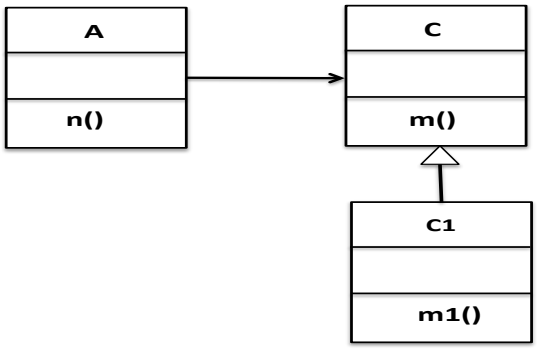
a) Adecuado porque es simple.
b) Inadecuado porque :usuario está sobrecargado.
c) Adecuado porque cada objeto sabe y realiza su tarea.
d) Inadecuado porque el objeto :trayecto solo representa un trayecto.

a) La simplicidad es un criterio de evaluación de un diseño, pero este diseño NO es simple porque la distribución de responsabilidades no se corresponde con los roles de cada clase. Por ejemplo, el objeto :usuario no se debe ocupar de pedir el trayecto ni de ordenar el cálculo de la factura.

b) Cierto. El objeto :usuario tiene más responsabilidades de las que se corresponden con su rol de manejar aquello referente al usuario. Es un ejemplo de la escoba… expuesto en clase.

c) En cualquier diseño cada objeto realiza la tarea asignada, salvo alguna equivocación. Por tanto, el argumento dado no sirve para evaluar lo adecuado de un diseño.

d) Es adecuado que el objeto :trayecto se ocupe de un solo trayecto.



8. En el diseño de la figura anterior:

a) El método m1() hereda a m() de C.
b) El método m() no existe en C1.
c) El método n() no puede utilizar el servicio m1().
d) El método m() existe en C1 pero es diferente a m() de C.

a) Falso. Un método NO hereda nada de nadie. La herencia es un mecanismo entre clases, no entre métodos.

b) Falso. El método m() existe en C1 aunque no sea visible en el diagrama porque lo hereda C1 de C.

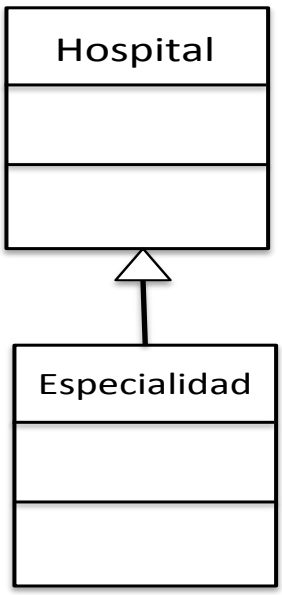
c) Cierto. El método n() no puede utilizar el servicio m1() porque no tiene acceso. Su relación es con la clase C y con la parte de la clase C1 que hereda de C, pero m1() es un método añadido a C1, no heredado de C. Por tanto, n() carece de acceso a m1().

d) Falso. El método m() de C1 es el mismo método m() de C, precisamente porque no se ha escrito en C1.

9. El diseño de la clase Coche es:

- **a) Adecuado porque sirve para dar los datos de un coche.**
- **b) Inadecuado porque solo representa a un coche.**
- **c) Adecuado porque contiene todos los "get" y "set" necesarios.**
- **d) Inadecuado porque solo es una fuente de datos.**

a) Falso. Es inadecuado dedicar una clase a dar datos.
b) Los objetos software no representan a elementos del universo exterior al software.
c) Contener los "get" y "set" no es condición de diseño adecuado.
d) Cierto. Es inadecuado dedicar una clase solo a contener datos. Los objetos son elementos activos, no almacenes de datos.



10. El diseño de la figura lateral es:

- **a) Correcto porque la especialidad está en el hospital.**
- **b) Correcto porque Especialidad es un caso particular de Hospital.**
- **c) Incorrecto porque Especialidad no deriva de Hospital.**
- **d) Incorrecto porque la flecha es al revés.**

a) Falso. La herencia software se interpreta solo en términos del verbo "ser"; "estar", no atañe a la herencia software.
b) Falso. Especialidad NO es un caso particular de un hospital.
c) Cierto. Especialidad no es un hospital,
d) Falso. No es una flecha; es un triángulo denominado discriminante.